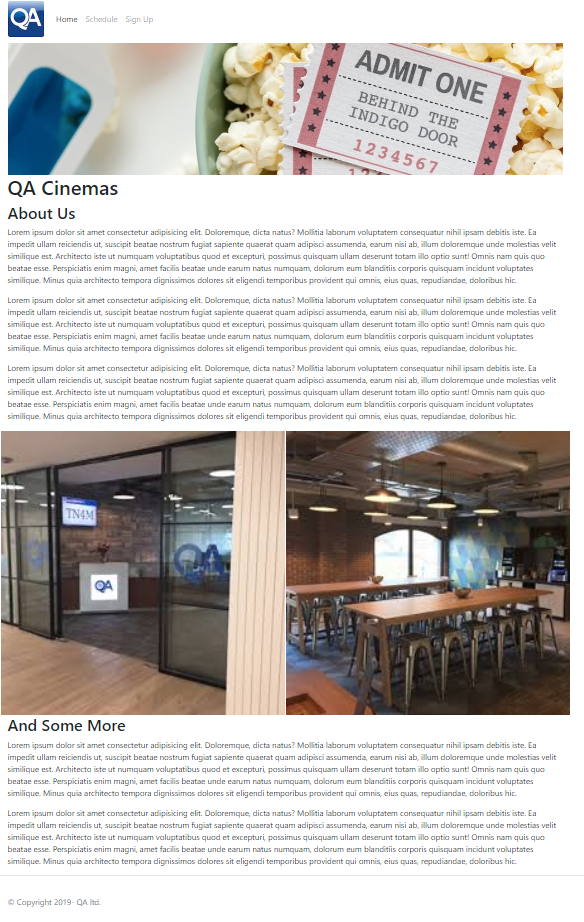
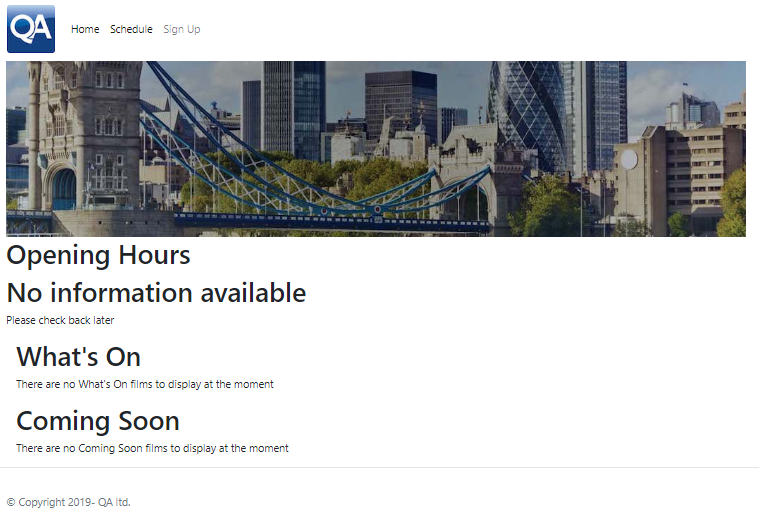
# QA Cinema Site

The web UI has already been written in React. Your job is to create an API service (written in C#) to provide the UI with appropriate services.

## Launching the React App

To see the react QA cinema website you need to:

1. Download the react logic from here: [QACS-TL/QACinemaProjectStarter (github.com)](https://github.com/QACS-TL/QACinemaProjectStarter)
2. Extract the zipped up logic into a folder of your choosing.
3. The extracted data includes a file called FilmsDB.sql which you can run in SQL Server Management System to create the Films database to be used by your C#.
4. start up an instance of Visual Studio Code.
5. Select Open Folder from the File menu and hunt down the folder called ReactQACinema that contains the react logic.
6. Select New Terminal from the Terminal window and point it at the folder that contains the package.json file. Then enter the following statements at the terminal (they will take a few minutes to execute):
   1. npm i
   2. npm start
7. Eventually a browser window should open which navigates to <https://localhost:3000> which will display the site’s landing page:  
   
8. Click on the website’s Schedule menu option. You should see the following page:  
     
   The opening hours, what’s on and coming soon sections are attempting to call on an API server that should produce appropriate information (in JSON format). This is the server you need to develop in C#. On completion the page should look something like the following:  
   Graphical user interface, application, website

   Description automatically generated
9. Users should be able to click on a film Title to see further information about a selected film:  
   Graphical user interface, text, application, email

   Description automatically generated
10. Click on the Sign Up menu and you should see the following:  
    Table

    Description automatically generated  
    On filling in the various controls the user should be able to click on the sign me up button and have their details stored in the application’s database

## API End-Points

The react app is listening on the following URLs for the various pieces of information:

**Opening times:**

https://localhost:7133/api/openings

Expects data in the following format:

[

{

"id": 1,

"day": "Monday ",

"opening": "11:00 ",

"close": "00:00 "

},

{

"id": 2,

"day": "Friday ",

"opening": "11:00 ",

"close": "2:00 "

},

{

"id": 3,

"day": "Sunday ",

"opening": "11:44 ",

"close": "23:00 "

},

{

"id": 4,

"day": "Tuesday ",

"opening": "14:30 ",

"close": "18:00 "

}

]

**Films by Status:**

Currently Showing Films (status = 1)

https://localhost:7133/api/films/filmsbystatus/1

Coming Soon (status = 2)

https://localhost:7133/api/films/filmsbystatus/1

Expects data in the following format:

[

{

"id": 1,

"title": "King of Thieves",

"synopsis": "Test Film 1 Synopsis",

"cast": "Test Film 1 Cast",

"directors": "Test Film 1 Directors",

"showingTimes": "11:45, 13:30, 17:15, 20:45",

"releaseDate": "2019-05-27T00:00:00",

"filmStatus": 1,

"image\_id": 1

},

{

"id": 2,

"title": "The Predator",

"synopsis": "Test Film 2 Synopsis",

"cast": "Test Film 2 Cast",

"directors": "Test Film 2 Directors",

"showingTimes": "11:45, 13:30, 17:15, 20:45",

"releaseDate": "2019-10-27T00:00:00",

"filmStatus": 1,

"image\_id": 2

},

{

"id": 4,

"title": "The Banana Men",

"synopsis": "Test Film 4 Synopsis",

"cast": "Test Film 4 Cast",

"directors": "Test Film 4 Directors",

"showingTimes": "9:25, 12:30",

"releaseDate": "2022-05-03T00:00:00",

"filmStatus": 1,

"image\_id": 4

}

]

**Film Images:**

https://localhost:7133/api/filmImages/id

Where id is an integer and the id of the selected film image

Expects data in the following format:

{

"id": 2,

"title": "The Predator",

"src": "images/ThePredator.jpg",

"alt": "The Predator"

}

Will return the following if the film has **NO** corresponding image:

{

"id": 4,

"title": "No Image",

"src": null,

"alt": "Image not available"

}

**Film specific details:**

<https://localhost:7133/api/films/id>

Where id is an integer and the id of the selected film

Expects data in the following format:

{

"id": 1,

"title": "King of Thieves",

"synopsis": "Test Film 1 Synopsis",

"cast": "Test Film 1 Cast",

"directors": "Test Film 1 Directors",

"showingTimes": "11:45, 13:30, 17:15, 20:45",

"releaseDate": "2019-05-27T00:00:00",

"filmStatus": 1,

"image\_id": 1

}

**Adding new Members:**

https://localhost:7133/api/members/

With new member details passed in the body of the request in the following format:

{

"id": "0",

"title": "Miss",

"firstName": "Alina",

"lastName": " Beatha",

"email": "Alina@ Beatha.com",

"phoneNumber": "01234567890",

"dob": "2022-05-31T00:00:00.000Z",

"gender": "female"

}

And returns the same but with the newly generated value in the Id field.

# Additional Information for the C# Project

The ASP NET API Server needs to have the following logic in its Program.cs file (new logic highlighted in bold):

var builder = WebApplication.CreateBuilder(args);

**var MyAllowSpecificOrigins = "\_myAllowSpecificOrigins"; //Needed for Cors**

// Add services to the container.

builder.Services.AddControllers();

**// Cors allows client requests to be made from same (localhost) machine**

**builder.Services.AddCors(options =>**

**{**

**options.AddPolicy(name: MyAllowSpecificOrigins,**

**policy =>**

**{**

**policy.WithOrigins("http://localhost:3000").AllowAnyHeader().AllowAnyMethod();**

**});**

**});**

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

**app.UseCors(MyAllowSpecificOrigins); //Needed for Cors**

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

The Cors logic added above is required to allow the C# project to accept requests from the react site that is being hosted on the same “localhost” server.